GRASP for the maximum diversity problem

Paulo Marcos F. de Andrade* Alexandre Plastino* Luiz Satoru Ochi*
Simone de L. Martins*

*Departamento de Ciência da Computação, Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brazil
pandrade@ic.uff.br,{plastino,satoru,simone}@dcc.ic.uff.br

1 Introduction

The maximum diversity problem (MDP) [2, 3, 4] consists of identifying optimally diverse subsets of individuals from populations. The selection of elements is made based on the diversity of their characteristics, calculated by a function applied on their attributes. The goal is to find the subset that presents the maximum possible diversity. There are many applications [6] that can be solved using the resolution of this problem, such as medical treatment, selecting jury panel, scheduling final exams and VLSI project.

This problem belongs to the class of NP-hard problems [3]. Glover et al. [3] present mixed integer zero-one formulation for this problem, that can be solved for small instances by exact methods. But this solution is not feasible for large problems due to large computational time.

Some heuristics and metaheuristics are available to obtain approximate solutions for this problem. Constructive and destructive heuristics were presented by Glover et al. [4]. Kochenberger et al. [6] showed results obtained using a tabu search and Katayama et al.[5] developed a memetic algorithm. Ghosh [2] proposed a GRASP that obtained good results for small instances of the problem.

In this paper, we present a new GRASP for this problem. In Section 2, we describe the construction and local search phases developed, and in Section 3, we present computational results for small and large instances of the problem. Concluding remarks are made in Section 4.

2 GRASP heuristic

Ghosh [2] proposed a GRASP heuristic for this problem and tested it for small instances. We developed a new GRASP and processed both of them for small and new large instances.

The GRASP [1] is an iterative process, where each iteration consists of two phases: construction and local search phase. The construction phase builds a feasible solution, whose

neighborhood is explored by local search. The best solution from all iterations is returned as result.The phases of both GRASPs are described below.

2.1 Construction phase

Let $S = \{s_i : i \in N\}, N = \{1, 2, \ldots, n\}$ be a population of n elements and s_{ik}, $k \in R = \{1, 2, ..., r\}$ the r values of the attributes of each element. The distance between any two elements i and j is the euclidean distance calculated as $d_{ij} = \sqrt{\sum_{k=1}^{r}(s_{ik} - s_{jk})^2}$. Let M be a subset of N and the overall diversity be $z(M) = \sum_{i<j:i,j\in M} d_{ij}$. The MDP problem consists of maximizing $z(M)$, subject to $|M| = m$.

The construction phase consists of m iterations. In the algorithm developed by Ghosh, for each iteration k, M_{k-1} is a partial solution with $k-1 (1 \leq k \leq m)$ elements. The element i^* to be inserted in each iteration is selected based on its contribution to the overall diversity $z(M)$. First, a lower bound $\Delta z_L(i)$ and an upper bound $\Delta z_U(i)$ are computed for all $i \in N - M_{k-1}$. Then a random number u is sampled from a uniform distribution $U(0, 1)$ that is used to compute $\Delta z'(i) = (1 - u)\Delta z_L(i) + u\Delta z_U(i)$. The selected element i^* is the one that presents the larger $\Delta z'$ and is included in M_{k-1} to obtain M_k. The computations of $\Delta z_L(i)$ and $\Delta z_U(i)$ are:

$\Delta z_L(i) = \sum_{j\in M_{k-1}} d_{ij} + \sum_{n-m+1\leq r\leq n-k} d_i^r(Q_{ik})$;

$\Delta z_U(i) = \sum_{j\in M_{k-1}} d_{ij} + \sum_{1\leq r\leq m-k} d_i^r(Q_{ik})$;

where $d_i^r(Q_{ik})$ is the rth largest distance in $\{d_{ij} : j \in Q_{ik}\}, Q_{ik} = N - M_{k-1} - \{i\}$.

The lower bound $\Delta z_L(i)$ is computed by adding the distances among i and the elements that are already in the solution to the sum of distances among i and the $m - k - 1$ elements that are not in solution M_{k-1} and which have smaller distances to i. The upper bound $\Delta z_U(i)$ is computed by adding the same first term of $\Delta z_L(i)$ to the sum of distances among i and the $m - k - 1$ elements that are not in solution M_{k-1} and which have larger distances to i.

We developed a new algorithm for the construction phase trying to improve the results generated by Ghosh's algorithm.

Initially, the vectors $SD(i) = \sum_{ji\in N} d_{ij}$ and $MD(i) = \sum_{j\in N} d_{ij}/n$ are computed for each element i. The first one is the sum of distances between i and the other elements and the second one is the average of these distances.

The initial element $i = i \in M_1$ is selected randomly from the m individuals that have larger values of SD, in order to start the construction of the solution with an element that presents a large sum of distances. Then for each iteration, a restricted candidate list is created and an element from this list is randomly selected. The algorithm to build the restricted list is in Figure 1.

In line 2, we compute for each element $i \in N - M_{k-1}$ the sum of distances $SDS(i)$ from it to the elements that are already in the solution. From lines 3 to 7, $\Delta z(i)$ is computed by two different ways for each element i. For the first $m/2$ elements to be inserted in the solution, we favour the elements that present an average of distances related to all population larger than

```
procedure Build_list (N, M_{k-1})
01.  for all i ∈ N − M_{k-1} do
02.    SDS(i) ← (Σ_{j∈M_{k-1}} d_ij) / (k−1);
03.    if ((SDS(i) > SD(i)) and (k > m/2)) then do
04.      Δz(i) ← SDS(i);
05.    else
06.      Δz(i) ← (SDS(i)+MD(i)) / 2;
07.    end if;
08.  end for;
09.  Sort i ∈ N − M_{k-1} by Δz;
10.  if m > n/2 then do
11.    lim_list ← n − m;
12.  else
13.    lim_list ← m;
14.  end if;
15.  D ← 0;
16.  for (i = 0, ..., lim_list − 1) do
17.    D ← D + (Δz(i) − Δz(i + 1));
18.  end for;
19.  D ← D/lim_list;
20.  Q ← 0;
21.  Restricted_List ← {∅};
22.  for (i = 1, ..., lim_list) do
23.    if ((Δz(i) − Δz(i + 1)) < D) then do
24.      Restricted_List ∪ {i}; Q ← Q + 1;
25.    else
26.      break;
27.  end for;
28.  return Restricted_List;
```

Figure 1: Algorithm for building the restricted candidate list

the distances related to the individuals of the partial solution M_{k-1}. For the selection of the next $m/2$ elements, the privileged individuals are the ones that present larger distances to the elements already in the partial constructed solution. In line 9, the elements that are not in the partial solution are sorted in decrescent order by their Δz values and in lines 10 to 14 we select an initial size for the initial restricted list. From lines 15 to 19, we compute the average of differences D among Δz of the elements from this list. From lines 20 to 27, we create the final restricted list by selecting elements from the initial list that present a difference among their Δz less than D, in order to assure that elements that present Δz close to another may be in the solution and to avoid the selection of elements that have Δz too distant from each other.

2.2 Local Search Phase

After a solution is constructed, a local search phase should be executed for attempting to improve the initial solution. The neighborhood of a solution defined by Ghosh [2] is the set of all solutions obtained by replacing an element in the solution by another that does not belong to the set of its elements. The incumbent solution M is initialized with the solution

obtained by the construction phase. For each $i \in M$ and $j \in N - M$, the improvement due to exchanging i by j, $\Delta z(i,j) = \sum_{u \in M_{\{i\}}} (d_{ju} - d_{iu})$ is computed. If for all i and j, $\Delta z(i,j) < 0$, the local search is terminated, because no exchange will improve z. Otherwise, the elements of the pair (i,j) that provides the maximum $\Delta z(i,j)$ are interchanged, a new incumbent solution M is created and the local search is performed again.

We use this same local search strategy for developing the GRASP presented here. The pseudo-code with the complete description of procedure `GRASP_newconstr` for the maximum diversity problem is given in Figure 2.

```
procedure GRASP_newconstr
01.  best_value ← 0;
02.  Compute SD(i) and MD(i) for all i ∈ N;
03.  Sort the elements in N by SD(i);
04.  for it = 1, ..., max_iterations do
05.    Choose randomly an individual from the first m elements of N to create M_1;
06.    for k = 2, ..., m do
07.      Restricted_List ← Build_List(N, M_{k-1});
08.      Select randomly i* from Restricted_List
09.      M_k = M_{k-1} ∪ {i*};
10.    end for
11.    best_weight = Σ_{i,j∈M_m} d_ij;
12.    do_local_search ← 1;
13.    while do_local_search do
14.      for all i ∈ M_m do
15.      Compute Δz(i,j), j ∈ M − {i};
16.      end for
17.      if max(Δz(i,j)) > 0 then do
18.        M_m = M_m − {i*} ∪ {j*}|Δz(i*,j*) = maxΔz(i,j);
19.        best_weight = best_weight + max(Δz(i,j));
20.      else do_local_search ← 0;
21.      end if
22.    end while
23.    if best_weight > best_value then do
24.      best_value ← best_weight;
24.    end if;
25.  end for;
```

Figure 2: Algorithm GRASP for MDP

3 Computational Results

We tested the proposed GRASP using two sets of problems. For the first one, we used small populations $n = 10$, $n = 20$, $n = 30$, $n = 40$ and $n = 50$ and implemented an exact algorithm that provided optimal solutions. This exact algorithm performed an exhaustive search in the solutions space. For the second one, we used populations of sizes $n = 100$, $n = 150$, $n = 200$, $n = 250$ and we were not able to run the exact algorithm due to CPU time limits.

Besides the size of population we had to specify the distances in the set $\{d_{ij}; i < j; i, j \in N\}$. We created four different sets for these distances: A, B, C and D. For the first one, we sampled

from a discrete uniform distribution over [1..9] the attributes for each individual and calculated d_{ij} as the euclidean distance of these attibutes. In the second one, we chosed randomly the distances d_{ij} from a uniform distribution over [1..9999]. For the third and fourth we sampled 50% of the elements from a uniform distribution over [1..9999] but the other 50% were sampled from a uniform distribution over [1..4999] for the first case, and over [5000..9999] for the second case.

The tests were processed using 10, 100 and 1000 iterations for each instance and each one was executed three times. We also implemented Ghosh algorithm to compare the results.

In Table 1, we show the results of computing 1000 iterations for the first set of tests (population 10, 20, 30, 40 and 50), for which we have optimal solutions. In the first line we identify the size of population. In the second line, we show the type of problem, that is the way the distances were computed. In the third line, we show the size of the subset to be selected (20% of the population). In the fourth and fifth columns, we present the optimality gap ($100[z_{exact} - z_{Grasp}]/z_{exact}$) for the average values(AS) obtained in three executions and the best solution (BS) found by Gosh algorithm. The sixth and seventh lines show the same results for our new algorithm. In the following lines we show the results obtained by Ghosh and our new algorithm for a subset of size equal to 40% of population.

We can see that the proposed GRASP found optimal solutions for all instances, while Ghosh algorithm could not find the optimal solution for instance B with population equal to 40 and subsite size equal to 16. Considering average solutions, we can see that the proposed algorithm missed to find the optimal solution only for one instance, while the Ghosh algorithm missed in two cases. The execution times for both algorithms were less than 1 second, while the exact algorithm took 12 hours of processing time.

For the second set of larger instances with population 100, 150, 200 and 250 we do not have the optimal solutions, so we considered the "optimal solution" for computing the optimality gap the best solution found by one of the algorithms executing 1000 iterations. The results are showed in Table 2.

Table 1: Results for both GRASPs compared with optimal solutions for 1000 iterations

Population		10				20				30				40				50			
Problem		A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
$\|m\|$		-				4				6				8				10			
Ghosh	AS	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BS	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
New Grasp	AS	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.08
	BS	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\|m\|$		4				8				12				16				20			
Ghosh	AS	0	0	0	0	0	0	0	0	0	0	0.13	0	0	0.15	0	0	0	0	0	0
	BS	0	0	0	0	0	0	0	0	0	0	0	0	0	0.15	0	0	0	0	0	0
New Grasp	AS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The proposed GRASP has found better results in eleven instances out of those fifteen where the two algorithms dont find the same solutions. We note also that the average solution corresponds to the better solution in 19 instances for the proposed GRASP and in 14 instances

Table 2: Results for both GRASPs compared with the best solution found for 1000 iterations

Population		100				150				200				250			
Problem		A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
$\|m\|$		20				30				40				50			
Ghosh	AS	0.01	0.09	0.08	0	0.02	0.26	0	0	0	0.30	0.06	0.03	0	0.13	0.13	0.07
	BS	0	0	0	0	0.01	0.13	0	0	0	0.30	0.06	0.03	0	0.04	0	0
New Grasp	AS	0	0	0	0	0	0	0	0.05	0	0.06	0	0.04	0	0.06	0.02	0.05
	BS	0	0	0	0	0	0	0	0.04	0	0	0	0	0	0	0	0.04
$\|m\|$		40				60				80				100			
Ghosh	AS	0	0	0	0.02	0	0.08	0	0	0	0.17	0.02	0.01	0	0.17	0.01	0
	BS	0	0	0	0.02	0	0.06	0	0	0	0.17	0.02	0	0	0.08	0	0
New Grasp	AS	0	0	0	0.01	0	0.03	0	0.01	0	0.08	0	0.02	0	0.02	0	0.06
	BS	0	0	0	0	0	0	0	0	0	0	0	0.01	0	0	0	0.05

for Ghosh algorithm.

The average computational time was quite similar for both algorithms ranging from 1s to 450s when the size of the subset is 20% of the population and from 1s to 3500s for a subset of 40% of the population.

4 Concluding Remarks

We described a GRASP algorithm for the maximum diversity problem, created new instances of the problem and compared the results obtained with optimal solutions and the solutions provided by the algorithm developed by Ghosh.

The tests showed that the GRASP heuristic is quite efficient for obtaing optimal solutions and good solutions, using much less time than an exact algorithm.

Comparing both GRASP algorithms, we can see that the GRASP proposed here give better solutions. The new construction phase gave better initial solutions and larger diversity of solutions that enabled the improvement of the solution by the local search phase in many cases.

We observed that 85% of the processing time for the proposed GRASP was due to the local search phase and, most of the times, it does not improve the initial solution. So we are investigating new local search strategies less intensive to improve the computational time while maintaining the quality of solutions.

References

[1] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptative search procedures", *Journal of Global Optimization* 6, 1995, 109-133.

[2] J. B. Ghosh, "Computational aspects of the maximum diversity problem", *Operations Research Letters* 19, 1996, 175-182.

[3] F. Glover, G. Hersh and C. McMillan, "Selecting subsets of maximum diversity", MS/IS Report No. 77-9, University of Colorado at Boulder, 1977.

[4] F. Glover, C-C. Kuo and K. S. Dhir, "Integer programming and heuristic approaches to the minimum diversity problem", *Journal of Business and Management*, 4(1), 1996, 93-111.

[5] K. Katayama and H. Naribisa, "An evolutionary approach for the maximum diversity problem", Working Paper, Dept. of Information and Computer Engineering, Okayama University of Science, 2003.

[6] G. Kochenberger and F. Glover, "Diversity data mining", Working Paper, Hearin Center for Enterprise Science, College of Business Administration,1999.